Priority-Based Human Resource Allocation in Business Processes*

Cristina Cabanillas[1], José María García[2], Manuel Resinas[3], David Ruiz[3], Jan Mendling[1], and Antonio Ruiz-Cortés[3]

[1]Vienna University of Economics and Business, Austria
{cristina.cabanillas,jan.mendling}@wu.ac.at
[2] STI Innsbruck, University of Innsbruck, Austria jose.garcia@sti2.at
[3]University of Seville, Spain {resinas,druiz,aruiz}@us.es

Abstract. In Business Process Management Systems, human resource management typically covers two steps: resource assignment at design time and resource allocation at run time. Although concepts like role-based assignment often yield several potential performers for an activity, there is a lack of mechanisms for prioritizing them, e.g., according to their skills or current workload. In this paper, we address this research gap. More specifically, we introduce an approach to define resource preferences grounded on a validated, generic user preference model initially developed for semantic web services. Furthermore, we show an implementation of the approach demonstrating its feasibility.
Keywords: preference modeling, preference resolution, priority-based allocation, priority ranking, RAL, resource allocation, SOUP

1 Introduction

Business Process Management System (BPMS) are increasingly used for supporting service composition. Typically, they work with executable process models that define the control flow, data processing, and resource involvement of a specific process. Resources in this context include both automatic services and services provided by human resources. In particular, the appropriate selection of human resources is critical as various factors such as workload or skills have an impact on work performance. While priorities for automatic services are intensively researched, it is surprising that prioritizing human resources has been hardly discussed. In classical workflow management, only two steps of resource management are considered: resource assignment at the level of process specification and resource allocation at run time [1]. *Resource assignment* builds on defining for each activity the human resources that are candidates to work on the activity. These are called *potential performers*. At run time, the *resource*

* This work was partially supported by the European Union's Seventh Framework Programme (FP7/2007-2013), the European Commission (FEDER), the Spanish and the Andalusian R&D&I programmes (grants 318275 (GET Service), 284860 (MSEE), TIN2009-07366 (SETI), TIN2012-32273 (TAPAS), TIC-5906 (THEOS)).

allocation step considers these potential performers to select *actual performers* (often a single person) who find the activity in their worklist. For instance, the Yet Another Workflow Language (YAWL) system [2] uses this concept to support various Workflow Resource Patterns (WRPs) [3]. Even though these WRPs identify strategies to balance workload, there is no explicit consideration of prioritizing potential performers to facilitate the selection of the actual performer. This is remarkable, as deriving a set of performers immediately poses the question of who would be the best candidate to pick up the work. Also professional solutions such as Activiti[1], WebSphere MQ[2] or BPEL4People, do not provide means to prioritize performers, but assign priority indicators to activities only.

In this paper, we address this research gap of how prioritization of resources can be integrated in BPMS. More specifically, we provide two contributions: (i) we conceptually define prioritized allocation based on preferences; and (ii) we propose a concrete way in which preferences over resources can be defined such that a resource priority ranking can be automatically generated. Our solution builds on the adaptation of a user preference model that was developed for the discovery and ranking of semantic web services [4] to the domain at hand. As a proof of concept, we have extended the resource management tool Collection of Resource-centrIc Supporting Tools And Languages (CRISTAL)[3] [5] with the Semantic Ontology of User Preferences (SOUP) component [6] to support priority-based allocation. In this system, Resource Assignment Language (RAL) [7] is used for resource selection.

The rest of the paper is structured as follows. In Section 2, we conceptually define priority-based allocation based on preferences and describe the requirements to address it based on a motivating scenario. In Section 3, we explain the adaptation of the preference metamodel and its formalization for automated prioritization. In Section 4, we evaluate the approach regarding preference modeling and resolution. Work related to preference modeling and resource prioritization in Business Processes (BPs) is detailed in Section 5, before closing the paper by drawing some conclusions and deriving potential future work in Section 6.

2 Priority-Based Resource Allocation

We define *priority-based resource allocation* as the ability to rank a set of resources according to one or more preferences. The result is thus a prioritized list of resources that can be allocated to a process activity. In the following, we present a real scenario that motivates the problem, and a set of requirements that must be considered in order to deal with it.

2.1 Motivating Scenario

The need for resource prioritization is motivated by a real scenario located in the Andalusian Institute of Public Administration (in Spanish IAAP), which

[1] `http://www.activiti.org/`
[2] `http://www-01.ibm.com/software/integration/wmq/`
[3] `http://www.isa.us.es/cristal`

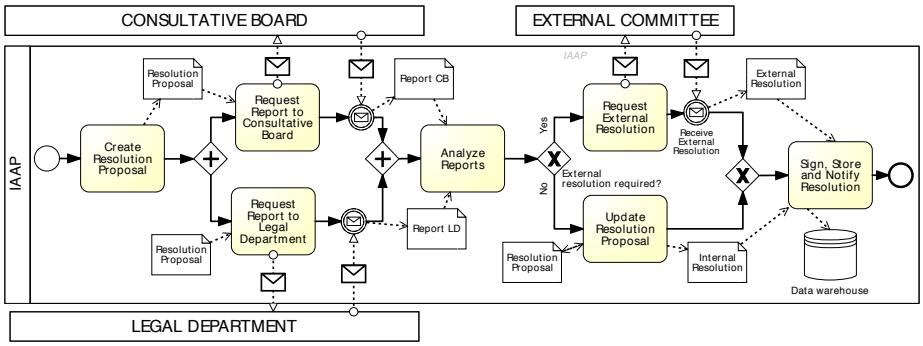

Fig. 1: BP to create and process a resolution proposal for hiring people

serves more than eight million end users. The Business Process (BP) in question represents the procedure to create and process a resolution proposal for hiring people and has a high use frequency in the IAAP. Fig. 1 shows the process in Business Process Model and Notation (BPMN). Once a draft of a resolution proposal is created, it is concurrently sent to the Consultative Board and to the Legal Department for evaluation. The IAAP then analyzes the reports and decides whether an external resolution is required. In that case, a request is sent to an external committee, which must create and send a new resolution. Otherwise, the resolution proposal is reviewed, and changes are applied to the initial one according to the reports received. In any case, the documents generated are signed and archived, and the resolution result is appropriately notified.

The part of the organizational structure of the IAAP related to Administrative Resource Management involved in this BP is hierarchically structured in eight *positions*: Business Manager (BM), Technician of the IAAP (T-IAAP), Assistant of the IAAP (A-IAAP), Secretary (SE), Assistant of the Legal Department (A-LD), Technician of the Legal Department (T-LD), Assistant of the Consultative Board (A-CB) and Technician of the Consultative Board (T-CB). They are occupied by a total of eleven people. Table 1 shows the selection conditions for each activity (positions are acronymized) and the preferences to prioritize the resulting set of potential performers for allocation.

2.2 Requirements for Resource Prioritization

The preferences for priority-based allocation as defined in the example have to yield a partial order relation over the set of potential performers of an activity. More specifically, we identify a set of requirements that must hold in order to deal with such automated resource prioritization as follows:

– *Resource assignment.* Some mechanism for resource assignment must be put into place to calculate the potential performers before resource prioritization. There are many different approaches for that purpose, e.g., [8–10].

Activity	**Potential Performers**	**Preferences**
Create Resolution Proposal	T-IAAP	The greatest number of past executions of the activity (*history*)
Request Report to CB/LD	T-IAAP	The best *balance* between low cost (*price*) and short worklist (*availability*), i.e., a person is preferred over another person if his/her cost is lower and his/her worklist is not longer, or *vice versa*
Analyze Reports	T-IAAP	The least average execution time for the activity (*speed*)
Request External Resolution	SE	The best skills on using a specific software application (*expertise*)
Update Resolution Proposal	T-IAAP, T-CB or T-LD	The same person who created the resolution draft (*Binding of Duties (BoD) [8]*)
Sign, Store and Notify Resolution	SE	A person that has been working for the company for at least one year (*experience*). In case of no distinction, the person with shortest worklist (*availability*)

Table 1: Resource selection conditions and resource preferences

- *Expressive preference modeling.* In this context, there is a need for expressive preferences that range from single-value criteria (e.g., age), to composite preferences where the preferences themselves can be ranked (cf. preference for *Sign, Store and Notify Resolution* in Table 1).
- *Preference resolution.* Some mechanism must enable the automated resolution of preferences at run time based on actual values.
- *Information availability.* In addition, similarly as for resource assignment, specific information (also called *properties*) about the resources needs to be stored, updated, and retrieved, to define and resolve resource preferences. For instance, the following types of properties can be distinguished for our motivating scenario:

  - *Personal and Organizational Data.* From personal data such as the ID number, name or age, to properties related to positions occupied in the company or functional roles being held. This data partially depends on the type of organizational model used in the company.
  - *Skills.* As shown for activity *Request External Resolution* in Table 1, knowledge on specific software applications, technologies, methodologies, *etcetera* may be of interest for resource prioritization.
  - *Professional Information.* Information on the salary of the resource (e.g. in terms of cost/hour) or the years of experience, can be necessary to determine the resource that best meets the needs of the company.
  - *Worklist.* The workload of the resources can also be critical to offer or allocate an activity to a specific person. An organization may prefer having balanced workloads rather than very busy people and idle people.

- *History.* For each person, information about the past execution of process activities must be stored, e.g. activities performed, BPs to which they belonged, execution time, number of times executed, *etcetera*.

This list cannot be exhaustive, but has to be adapted for the context of the process at hand. For instance, in other cases it might also be necessary to access calendar data, such as scheduled meetings or holidays, in order to prioritize according to availability on a specific date.

3 Materializing Priority-Based Resource Allocation

As derived from the previous section, the main challenge for resource prioritization is two-fold: (i) to come up with a mechanism to express preferences over resources, and (ii) to develop a way to rank a set of resources according to those preferences. Furthermore, the solution should be as independent as possible from the properties used in the preferences so that they can accommodate the different requirements several organizations may have.

The first challenge is solved by leveraging SOUP, a highly expressive user preference model defined by García et al. [4, 6] that we adapt to the BP domain in this paper. To deal with the second challenge, we have developed a novel algorithm to rank resources according to the preferences expressed in SOUP. Both the preference metamodel and the algorithm are independent of the properties used in the preferences.

3.1 SOUP: A Metamodel to Define Preferences

In SOUP, a preference can be intuitively expressed as "I prefer y rather than x", where x and y are instances of domain concepts that represent properties of the resources to be allocated (e.g., size of the worklist). This relation between concept instances can be mathematically interpreted as a strict partial order. Therefore, in SOUP a preference can be defined as:

Definition 1 (Preference). *Let $\mathcal{C}$ be a non-empty set of domain concepts, and* $\mathrm{dom}(\mathcal{C})$ *the set of all possible instances of those concepts. We define a preference as $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$, where $<^{\mathcal{P}} \subseteq \mathrm{dom}(\mathcal{C}) \times \mathrm{dom}(\mathcal{C})$ is a strict partial order (irreflexive, transitive and asymmetric), and if $x, y \in \mathrm{dom}(\mathcal{C})$, then $x <^{\mathcal{P}} y$ is interpreted as "I prefer* y *rather than* x*".*

Consequently, each preference term instance defines its order depending on the concrete concepts referred ($\mathcal{C}$) and some operand values that determine the evaluation of the $<^{\mathcal{P}}$ relation. Furthermore, if we consider a finite set of concept instance pairs $(x, y) \in <^{\mathcal{P}}$, $\mathcal{P}$ can be represented as a directed acyclic graph, also known as *Hasse diagram* [11], where each node corresponds to a concept instance, and edges represent the preference relation $<^{\mathcal{P}}$.

Fig. 2 shows a UML representation of SOUP preference terms. Atomic preferences can be expressed using different preference terms, whereas composite

preferences can be used to compose those terms, defining the relation between previously expressed atomic preferences. Both atomic and composite preferences are handled by ranking mechanisms that implement the ranking process according to the corresponding term definition.

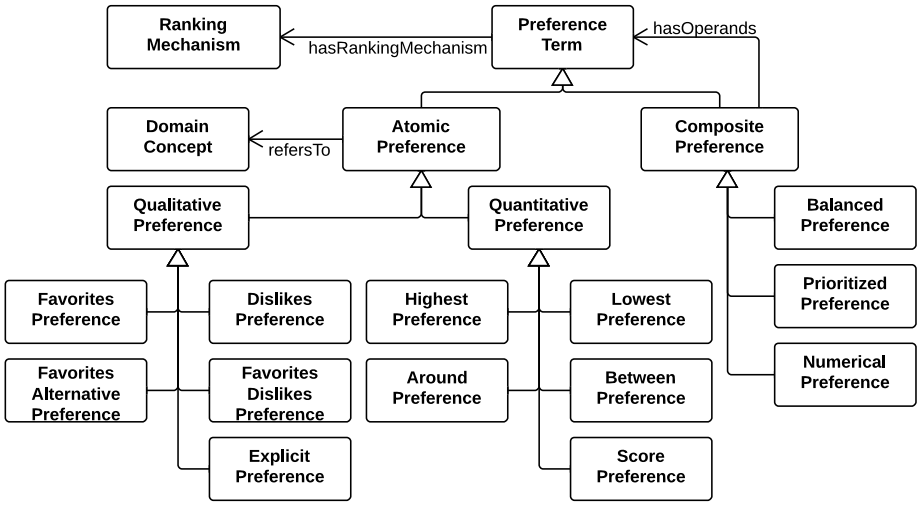

Fig. 2: UML representation of SOUP

In particular, atomic preferences are related to a domain-specific concept that usually represents a property that should be optimized to fulfill the user preference over it. SOUP supports both qualitative and quantitative preferences, depending on the nature of the property referred by the concrete preference. On the one hand, if the property is qualitative, e.g. the skills of a resource, one can use a `Favorites` preference to state that certain values of that property are favored against the rest (e.g., skills on a concrete software application). Conversely, a `Dislikes` preference can be used to enumerate the values that should not be provided for the referred property. A `FavoritesAlternative` allows defining a favorite and an alternative set of property values, meaning that values contained in the former set are the most preferred, but if there is none then values from the latter set can also be considered. A `FavoritesDislikes` preference is a combination of a `Favorites` and a `Dislikes` preference, where preferred values are the ones in the favored set or at least not in the disfavored set. Finally, an `Explicit` preference simply states the preference between two concrete values of a property (e.g. skills on LibreOffice are more preferred than skills on Microsoft Office).

On the other hand, quantitative preferences compare numerical values of the related properties. Thus, a `Highest` (`Lowest`) preference means the user prefers higher (lower) values for the referred property. `Around` and `Between` preferences

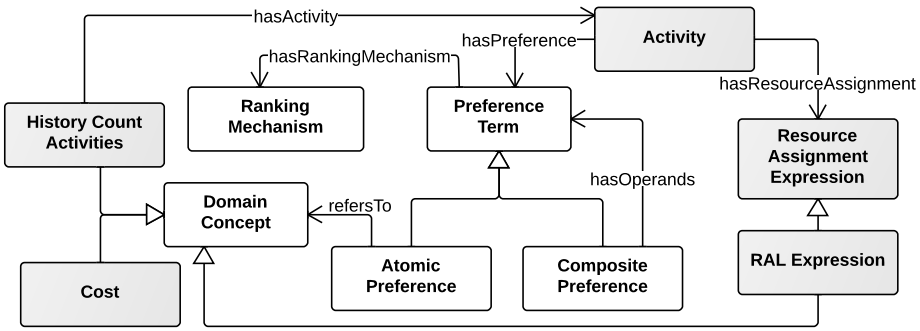

Fig. 3: Modeling priority-based resource allocation using SOUP

prefer values that are close to a specific value, or included within an interval, respectively. Finally, a `Score` preference expresses that the property value will be evaluated using a scoring function that returns a real value between 0 and 1 expressing to what extent the referred property value is preferred against others.

Concerning composite preferences, SOUP provides three facilities for defining the semantics. First, a `Balanced` preference $\mathcal{P}$ combines preference terms $\mathcal{P}_1, \ldots, \mathcal{P}_n$ using the *Pareto-optimality principle*. Therefore, a resource rsc_l is considered better than another resource rsc_m with respect to $\mathcal{P}$ if rsc_l is better than rsc_m with respect to any P_i such that rsc_l is not worse than rsc_m with respect to the rest of the combined preferences $\mathcal{P}_j$ with $i \neq j$.

Second, a `Prioritized` preference combines preferences in importance order. If a list of terms $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$ is combined using this operator, resources will be ranked first in terms of $\mathcal{P}_1$. Those that are equally preferred using $\mathcal{P}_1$ are ranked in terms of $\mathcal{P}_2$, and so forth.

Finally, a `Numerical` preference is the combination of preferences using a real function to obtain a numerical score value for each resource. Resources are ranked in terms of their score values. However, this composite preference can only combine quantitative preferences that can be evaluated to a score value.

3.2 Modeling Priority-Based Resource Allocation with SOUP

Two aspects must be modeled for resource prioritization, namely resource assignments for defining potential performers, and preference modeling for defining their order of priority (cf. Fig. 3).

First, each activity of the BP has a resource assignment expression that defines its set of potential performers. In our case, we use RAL [12] as a language to select a set of potential performers. The language is grounded on a consensual organizational metamodel [13] that takes into consideration not only people and roles, but also positions, organizational units and skills. It also allows selecting people based on the performers of previous activities in the process. We have chosen RAL for its expressiveness, extensibility, and for its capabilities to automatically resolve RAL expressions [7].

Second, preferences for resource prioritization are formulated using SOUP `Preference Term`s. All such preferences in SOUP must refer to some domain concept that represents the properties used in the preference. Therefore, modeling the preferences specified for each of the activities of the scenario also involves identifying the domain concepts used to prioritize resources. For instance, our scenario requires domain concepts like `Cost` and the number of times a resource has carried out a certain activity (`History Count Activities`). Additionally, a RAL expression can also be used in place of a domain concept. An example of such a preference is *I prefer the resource that is responsible for activity Create Resolution Proposal*, which can be modeled by means of a preference term of type `Favorites` referred to a domain concept of type `RAL Expression` and an operand that specifies the expression: `IS PERSON RESPONSIBLE FOR ACTIVITY Create Resolution Proposal`.

3.3 Ranking Resources According to SOUP Preferences

The prioritization of resources according to preferences is based on ranking mechanisms, which can be defined as follows:

Definition 2 (Ranking mechanism). *We define a ranking mechanism as an algorithm that receives as input a preference and a set of resources to be ranked, and returns as output a partially ordered set of the resources ranked according to the preference.*

Each `Preference Term` has a ranking mechanism that is used to implement the ranking process according to the corresponding term definition. Ranking mechanisms can be shared between different preference terms. In [6], ranking mechanisms for composite preference terms are discussed. However, the authors do not provide details about the ranking mechanisms for atomic preferences because they heavily depend on the characteristics of the knowledge base that contains the information about the domain concepts used in the preferences.

Therefore, in this section we adapt SOUP by providing a formalization of a generic ranking mechanism for atomic preferences, which has been designed to deal with the following characteristics of the knowledge base used for priority-based resource allocation: (1) it is usually distributed in different heterogeneous repositories such as a process log, an organizational model and an Enterprise Resource Planning (ERP) system; (2) it is dynamic in the sense that some of its repositories are continously changing (e.g., the worklist of each resource); and (3) computing the value of a domain concept for a resource can be a complex operation. For instance, computing the availability of a resource may involve checking his/her agenda, a calendar of the holidays of the country in which s/he works and the worklist of his/her pending tasks.

Reflecting these characteristics, we separate the evaluation of a domain concept for a resource from the ordering of the resources according to this evaluation. The first task is solved by defining a function $eval_{\mathcal{P}}^{KB}$ that is specific for each domain concept and for each knowledge base and can be defined as follows.

Definition 3 (Domain concept evaluation). *Let KB be a knowledge base, $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$ be an atomic preference, and $\mathcal{R}$ be the set of resources of the organization.*

- *If the domain concept $\mathcal{C}$ represents a quantitative property, then we define the evaluation as a function $eval^{KB}_{\mathcal{P}} : \mathcal{R} \to \mathbb{R}$, such that it returns a real number that represents the value of the domain concept for the given resource.*
- *If the domain concept $\mathcal{C}$ represents a qualitative property, then we define the evaluation as a function $eval^{KB}_{\mathcal{P}} : \mathcal{R} \to \{false, true\}$, such that it returns a boolean that represents whether the value of the domain concept belongs to the set specified in the preference.*

For instance, the evaluation of the quantitative concept *size of the worklist* for a resource r is the size of the worklist of resource r. Similarly, the evaluation of the qualitative concept *resource expression* with operand *IS PERSON RESPONSIBLE FOR ACTIVITY Create Resolution Proposal* (which is a RAL expression) for a resource r is true if, according to the organizational model, r is a person responsible for that activity.

The second task that must be performed involves applying the partial order specified by the type of a preference to the result of an evaluation. This is done by defining for each type of preference, a function that compares these results.

Definition 4 (Greater-than comparison). *Let KB be a knowledge base with information about the domain concepts used in the preferences, $\mathcal{P}$ be an atomic preference such that $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$, and D be the range of function $eval^{KB}_{\mathcal{P}}$ for such preference. We define a greater-than comparison for partial order $<^{\mathcal{P}}$ as a function $gt_{<^{\mathcal{P}}} : D \times D \to \{false, true\}$ such that $gt_{<^{\mathcal{P}}}(d_i, d_j)$ returns true if d_i is greater than d_j according to $<^{\mathcal{P}}$, i.e., if d_i is preferred over d_j.*

The implementation of this function can be straightforwardly derived from the type of the preference it corresponds to. For instance, if $<^{\mathcal{P}}$ corresponds to the type *Higher*, then $gt_{\text{Higher}}(n, n')$ is true iff $n > n'$. The same procedure can be applied to all of the other types of atomic preferences.

Using these two functions it is easy to provide an implementation of a generic ranking mechanism of atomic preferences as depicted in Algorithm 1. The algorithm simply iterates over the list of resources provided to the ranking mechanism and adds an edge $r \to r'$ to the graph if r' is preferred to r according to the preference $\mathcal{P}$ and the information in the knowledge base KB. Note that it is possible to provide more efficient implementations by means of domain-specific ranking mechanisms that leverage capabilities of the repositories of the knowledge base. However, a discussion of the different implementations of ranking mechanisms that can be developed is out of the scope of this paper.

Regarding ranking mechanisms for composite preferences, a similar approach could be followed. For instance, a ranking mechanism for composite preference `Prioritized` could be implemented using the same algorithm as Algorithm 1 but changing function $gt_{<^{\mathcal{P}}}$ for a function that uses the ranking mechanisms of its composed preferences to obtain partial results and, then, composes all of them together according to the semantics of `Prioritized`.

Algorithm 1 Generic ranking mechanism of atomic preferences

IN: A knowledge base KB, an atomic preference $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$ and a set of potential performers $\mathcal{R}$
OUT: A strict partially ordered set of resources $POSET$
add all resources $\mathcal{R}$ as nodes of $POSET$
for all $r \in \mathcal{R}$ **do**
  for all $r' \in \mathcal{R} \setminus \{r\}$ **do**
    add edge $r \rightarrow r'$ to $POSET$ if $gt_{<^{\mathcal{P}}}(eval_{\mathcal{P}}^{KB}(r'), eval_{\mathcal{P}}^{KB}(r))$
  end for
end for

Finally, having all of these ranking mechanisms, the priority-based allocation for an activity A of a BP just involves using the appropriate ranking mechanism with the allocation preferences of such an activity and the set of resources selected during the resource assignment as inputs. Then, from the partially ordered set obtained by the ranking mechanism it is easy to derive a total order that is a linear extension of the partial order using well-known topological sorting algorithms [14]. Note that preferences define a strict partial order amongst the resources. This means that given two resources r_1 and r_2 it may not be possible to establish a preference between them. In that case, the resources are randomly ordered.

4 Evaluation

To validate our proposal, we have developed a proof-of-concept implementation[4] for priority-based allocation and we have applied it to the scenario detailed in Section 2.1. The implementation can be divided into a domain-independent part and a domain-specific part. The first part is based on PURI [6], a preference framework that provides the building blocks to implement new ranking mechanisms and also provides an implementation of the ranking mechanisms of all of the composite preferences. For our proof-of-concept, we implemented the generic ranking mechanism for both the atomic preferences and for all *gt* functions based on the preference types defined in SOUP. The second part involves two steps: (1) identifying the domain concepts used to prioritize resources and modeling the preferences for each activity of the process, and (2) identifying the knowledge base and implementing the *eval* functions to evaluate the domain concepts. We discuss these two steps in the following.

4.1 Modeling the Preferences

Modeling the preferences specified for each of the activities of the scenario involves identifying the domain concepts used to prioritize resources, and using the SOUP metamodel to express the preferences regarding such domain concepts.

[4] Available at `http://www.isa.us.es/cristal`

```
Request Report to CB/LD: Assigned to persons with position T-IAAP.
The preference is balanced between low cost and short worklist.
iaap:RequestReport a bp:Activity ;
   bp:hasResourceAssignment [
      a ral:Expression ;
      ral:expr "HAS POSITION T-IAAP"
   ] ;
   bp:hasPreference [
      a soup:BalancedPreference ;
      soup:hasOperands
         [ a soup:LowestPreference ; soup:refersTo org:cost ] ,
         [ a soup:LowestPreference ; soup:refersTo worklist:size ]
   ] .

Update Resolution Proposal: Assigned to people with positions T-IAAP,
T-CB or T-LD. The preferred person is that who did the initial proposal.
iaap:UpdateProposal a bp:Activity ;
   bp:hasResourceAssignment [
      a ral:Expression ;
      ral:expr "HAS POSITION T-IAAP OR HAS POSITION T-CB OR
                HAS POSITION T-LD"
   ] ;
   bp:hasPreference [
      a soup:FavoritesPreference ;
      soup:refersTo ral:Expression ;
      soup:hasFavorites "IS PERSON RESPONSIBLE FOR ACTIVITY
                         Create Resolution Proposal Draft"
   ] .
```

Fig. 4: Examples of preferences expressed in RDF/Turtle syntax

Due to space constraints, we illustrate how these two tasks have been done with regard to activities *Request Report to CB/LD* and *Update Resolution Proposal*. The same approach can be used with the remaining activities.

Fig. 4 depicts the resource assignment (`bp:hasResourceAssignment`) and the preferences (`bp:hasPreferences`) for the aforementioned activities expressed in SOUP using RDF/Turtle syntax. For activity *Request Report to CB/LD* the preference is composite because it balances two atomic preferences: lowest cost and shortest worklist. Therefore, the composite preference is represented by means of an element of type `soup:BalancedPreference`, which is the type of composite preference that best suits the intention of the modeler. Regarding the atomic preferences, they are connected with the composite preference by means of relation `soup:hasOperands`. Both atomic preferences are of the same type (soup:LowestPreference). However, the former refers to the domain concept expressed with `org:cost`, which represents the cost of the resource, whereas the latter refers to the domain concept expressed with `worklist:size`, which represents the size of the worklist of the resource. Note that these two domain

concepts must have an *eval* function implemented for them so that they can be evaluated for each resource.

The preference for activity *Update Resolution Proposal* is also atomic. However, it refers to the organizational information stored about each resource. In particular, it sets a qualitative preference stating that it prefers the people that fulfill the condition stated by property `soup:hasFavorites`.

4.2 Identifying the KB and Implementing *eval* Functions

In our scenario, there are three repositories that store the values of the properties used for the prioritization of resources. The *organizational repository* stores personal information of all members of the company along with their positions, roles, and units within the organization, information about skills, salary or hiring date, and all the data that is not related to participation in BP activities. The *worklist repository* stores the worklists of all resources in the organization. Finally, the *history repository* stores the event log of past process executions. The organizational repository is manually updated, whereas the worklist repository and the history repository are updated by the BPMS, which in our implementation is Activiti. The *eval* functions can be divided into the following four categories according to the repository they use to evaluate the domain concept:

Quantitative organizational evaluations: They are used to evaluate concepts that represent quantitative properties stored in the organizational repository such as personal (e.g., age) or professional information (e.g., salary).

Qualitative organizational evaluations: They are used to evaluate concepts that represent qualitative properties stored in the organizational repository including organizational information and skills. The implementation of this type of evaluations leverages CRISTAL [5] to resolve RAL expressions and obtain the set of resources.

Worklist evaluations: They are used to evaluate concepts that represent quantitative properties related to the size of the worklist. They interact with the BPMS to obtain the information about them.

History evaluations: They are used to evaluate concepts that represent quantitative properties about the participation of resources in past process activity executions. This evaluation also accesses the history of the BP.

5 Related Work

We next present a summary of the pros and cons of the current approaches for preference modeling in different domains, followed by an analysis of current support for resource prioritization in Business Process Management (BPM).

Preference Modeling. There are several formalisms that can be used to represent preferences in different fields [15]. Quantitative preferences modeled as utility or scoring functions have been widely used in economics and operations research [16,

17], as well as in web systems [18, 19]. This approach solves the multiple criteria decision making by transforming it to aggregated scoring functions. However, these functions are difficult to define by users, and not all the preferences that are strict partial orders can be represented [20]. In artificial intelligence research, solutions have been focused on defining preferences in a qualitative way, easier to understand and more natural to define by humans [21]. These preference models offer facilities to define preferences as a set of statements or terms that are contextually related. In database research, there are also several solutions, for instance, using top-k or skyline algorithms to obtain the best search items according to a stated preference [22, 20]. These preference models usually offer qualitative facilities to define preferences, though their implementation usually leads to large result sets that do not discriminate well between items to be ordered [15]. We have chosen SOUP [4] because it is a hybrid approach, as it combines quantitative and qualitative facilities to define preferences. Furthermore, it is independent of the domain, so it is suitable for resource prioritization. Indeed, the expressiveness of the model is semantically close to BP modeling, which enables its interoperability with other resource allocation solutions.

Resource Prioritization in Business Processes. Regarding the prioritization of resources in BPM, we have studied the support provided by the specifications BPMN 2.0, BPEL4People and WS-HumanTask; the BPMS Activiti and YAWL [10]; and the product suites WebSphere MQ and ARIS systems, concerning preference definition and resource ranking. We find that they neither provide support for preference specification nor for ranking of potential performers in order to prioritize resource allocation. Crowdsourcing systems, which outsource the execution of activities to the crowd, usually rely on a fixed set of features such as skills, location, certification, cost or reputation to implement priotization [23], or they use some auction or competition mechanism to select the best worker [24]. However, the prioritization mechanism is defined in the system for all the activities and cannot be customized according to other criteria.

The importance of ranking resources is also emphasized in other recent work. In [25], the authors define a resource visualization concept that is aimed to support resource allocation using three different metrics to recommend work distribution. The distribution of work is addressed in [26] trying to keep the balance between quality and performance. In [27], six resource allocation mechanisms are compared with regard to suitability, urgency, conformance and availability. Although all of them agree on the need of dynamic work allocation to adapt to the evoluting needs of organizations, they do not deal with preference modeling itself. Some other solutions approach this problem from a process mining perspective, focusing on providing recommendation from information inferred from event logs [28–31]. A resource manager is finally responsible for making the final decisions for allocation. Altogether, the work presented in this paper generalizes and complements the current support in the field of BPM regarding preference modeling. It would be interesting to combine it, e.g., with visualization support as proposed in [25, 31].

6 Conclusions and Future Work

In this paper, we addressed the problem of integrating priorities into resource assignment and allocation. To this end, we extended concepts from preference modeling and combined them with resource management techniques.

The main advantage of our solution is that it provides a mechanism to define a wide variety of different types of preferences while being independent of both the properties used in the preferences and the knowledge base that contains the information about them. These features make it easier to accommodate the different requirements several organizations may have. This is a significant difference with respect to the support provided by current systems, which are defined to deal with a specific set of properties. Furthermore, the expressiveness of the preferences that can be defined with our approach outperforms the current support in the BPM field regarding priority-based allocation.

We plan to extend the preference model to support complex cases involving the agenda of the resources in order to allow expressing and using preferences referring to the expected end time of an activity. We also want to explore how a similar technique could be applied to the distribution of work to resources, i.e., in the opposite direction. Activity priority might be considered in that case.

References

1. N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and D. Edmond, "Workflow Resource Patterns: Identification, Representation and Tool Support," in *CAiSE*, pp. 216–232, 2005.
2. W. M. P. van der Aalst and A. H. M. ter Hofstede, "YAWL: Yet Another Workflow Language," *Inf. Syst.*, vol. 30, no. 4, pp. 245–275, 2005.
3. H. Tan and W. M. P. van der Aalst, "Implementation of a YAWL Work-List Handler based on the Resource Patterns," in *CSCWD'06*, pp. 1–6, 2006.
4. J. M. García, D. Ruiz, and A. Ruiz-Cortés, "A Model of User Preferences for Semantic Services Discovery and Ranking," in *ESWC (2)*, vol. 6089 of *LNCS*, pp. 1–14, Springer, 2010.
5. C. Cabanillas, A. del Río-Ortega, M. Resinas, and A. Ruiz-Cortés, "CRISTAL: Collection of Resource-centrIc Supporting Tools And Languages," in *BPM 2012 Demos*, vol. 940, pp. 51–56, 2012.
6. J. M. García, M. Junghans, D. Ruiz, S. Agarwal, and A. Ruiz-Cortés, "Integrating Semantic Web Services Ranking Mechanisms Using a Common Preference Model," *Knowledge-Based Systems*, vol. 49, pp. 22–36, 2013.
7. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "Defining and Analysing Resource Assignments in Business Processes with RAL," in *ICSOC*, vol. 7084, pp. 477–486, 2011.
8. M. Strembeck and J. Mendling, "Modeling process-related RBAC models with extended UML activity models," *Inf. Softw. Technol.*, vol. 53, pp. 456–483, 2011.
9. A. Awad, A. Grosskopf, A. Meyer, and M. Weske, "Enabling Resource Assignment Constraints in BPMN," tech. rep., BPT, 2009.
10. M. Adams, "The Resource Service," in *Modern Business Process Automation*, pp. 261–290, 2010.

11. B. A. Davey and H. A. Priestley, *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press, 2002.
12. C. Cabanillas, M. Resinas, and A. Ruiz-Cortés, "RAL: A High-Level User-Oriented Resource Assignment Language for Business Processes," in *Business Process Management Workshops (BPD'11)*, pp. 50–61, 2011.
13. N. Russell, A. ter Hofstede, D. Edmond, and W. M. P. van der Aalst, "Workflow Resource Patterns," tech. rep., BETA, WP 127, Eindhoven University of Technology, 2004.
14. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2001.
15. C. Domshlak, E. Hüllermeier, S. Kaci, and H. Prade, "Preferences in ai: An overview," *Artif. Intell.*, vol. 175, no. 7-8, pp. 1037–1052, 2011.
16. P. C. Fishburn, *Utility theory for decision making*. Wiley, 1970.
17. R. L. Keeney and H. Raiffa, *Decisions with multiple objectives: Preferences and value tradeoffs*. Cambridge Univ Press, 1993.
18. R. Agrawal and E. L. Wimmers, "A Framework for Expressing and Combining Preferences," in *SIGMOD Conference*, pp. 297–306, 2000.
19. L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 311–327, 2004.
20. J. Chomicki, "Preference formulas in relational queries," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 427–466, 2003.
21. C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements," *J. Artif. Intell. Res. (JAIR)*, vol. 21, pp. 135–191, 2004.
22. W. Kießling, "Foundations of Preferences in Database Systems," in *VLDB*, pp. 311–322, 2002.
23. M. Vukovic, "Crowdsourcing for Enterprises," in *SERVICES*, pp. 686–692, 2009.
24. B. Satzger, H. Psaier, D. Schall, and S. Dustdar, "Auction-based crowdsourcing supporting skill management," *Inf. Syst.*, vol. 38, no. 4, pp. 547–560, 2013.
25. M. De Leoni, M. Adams, W. M. P. van der Aalst, and A. H. M. Ter Hofstede, "Visual support for work assignment in process-aware information systems: Framework formalisation and implementation," *Decis. Support Syst.*, vol. 54, no. 1, pp. 345–361, 2012.
26. A. Kumar, W. M. P. van der Aalst, and E. M. W. Verbeek, "Dynamic Work Distribution in Workflow Management Systems: How to Balance Quality and Performance," *J. Manage. Inf. Syst.*, vol. 18, no. 3, pp. 157–193, 2002.
27. H. A. Reijers, M. H. Jansen-Vullers, M. Z. Muehlen, and W. Appl, "Workflow management systems + swarm intelligence = dynamic task assignment for emergency management applications," in *BPM*, pp. 125–140, 2007.
28. T. Liu, Y. Cheng, and Z. Ni, "Mining event logs to support workflow resource allocation," *Know.-Based Syst.*, vol. 35, pp. 320–331, 2012.
29. Y. Liu, J. Wang, Y. Yang, and J. Sun, "A semi-automatic approach for workflow staff assignment," *Comput. Ind.*, vol. 59, no. 5, pp. 463–476, 2008.
30. S. Rinderle-Ma and W. M. P. van der Aalst, "Life-Cycle Support for Staff Assignment Rules in Process-Aware Information Systems." Department of Technology Management, Eindhoven University of Technology, 2007.
31. R. P. J. C. Bose and W. M. P. van der Aalst, "Process Mining Applied to the BPI Challenge 2012: Divide and Conquer While Discerning Resources," in *BPM Workshops*, pp. 221–222, 2012.